

**IBM NALAIYA THIRAN PROFESSIONAL READINESS FOR INNOVATION,  
EMPLOYABILITY AND ENTREPRENEURSHIP**

**PERSONAL EXPENSE TRACKER**

**A PROJECT REPORT**

**Submitted by**

**AJITESH M - 2019103503**

**DEEKSHITH M – 2019103014**

**VISHNUPRIYA N – 2019103599**

**HEMANTH D – 2019103020**

**ESWARAMOORTHY B - 2019103017**

**COLLEGE OF ENGINEERING, GUINDY**

**ANNA UNIVERSITY : CHENNAI 600 025**

# TABLE OF CONTENTS

CHAPTERNO.	TITLE
	<b>ABSTRACT</b>
	<b>LIST OF FIGURES</b>
<b>1</b>	<b>INTRODUCTION</b> <ul style="list-style-type: none"><li>1.1Project Overview</li><li>1.2Purpose</li></ul>
<b>2</b>	<b>LITERATURE SURVEY</b> <ul style="list-style-type: none"><li>2.1Existing problem</li><li>2.2References</li><li>2.3Problem Statement Definition</li></ul>
<b>3</b>	<b>IDEATION &amp; PROPOSED SOLUTION</b> <ul style="list-style-type: none"><li>3.1Empathy Map Canvas</li><li>3.2Ideation &amp; Brainstorming</li><li>3.3Proposed Solution</li><li>3.4Problem Solution fit</li><li>3.5 Solution Architecture</li></ul>
<b>4</b>	<b>REQUIREMENT ANALYSIS</b> <ul style="list-style-type: none"><li>4.1Functional requirement</li><li>4.2Non-Functional requirements</li></ul>
<b>5</b>	<b>PROJECT DESIGN</b> <ul style="list-style-type: none"><li>5.1Data Flow Diagrams</li><li>5.2Technical Architecture</li><li>5.3User Stories</li><li>5.4 Customer Journey Map</li></ul>
<b>6</b>	<b>PROJECT PLANNING &amp; SCHEDULING</b> <ul style="list-style-type: none"><li>6.1Sprint Planning &amp; Estimation</li><li>6.2Sprint Delivery Schedule</li></ul>

6.3Reports from JIRA

<b>7</b>	<b>CODING &amp; SOLUTIONING</b>
	7.1Setting Limit
	7.2Sending Alert mail
<b>8</b>	<b>TESTING</b>
	8.1Test Cases
	8.2User Acceptance Testing
<b>9</b>	<b>RESULTS</b>
	9.1Performance Metrics
<b>10</b>	<b>ADVANTAGES &amp; DISADVANTAGES</b>
<b>11</b>	<b>CONCLUSION</b>
<b>12</b>	<b>FUTURE SCOPE</b>
<b>13</b>	<b>APPENDIX</b>
	Source Code
	GitHub & Project Demo Link

## LIST OF FIGURES

<b>FIGURENO.</b>	<b>TITLE</b>
<b>1.1</b>	<b>Empathy Map</b>
<b>1.2</b>	<b>Problem Statement &amp; Brainstorm</b>
<b>1.3</b>	<b>Group Ideas &amp; Prioritize</b>
<b>1.4</b>	<b>Problem Solution fit</b>
<b>2.1</b>	<b>Data Flow Diagram</b>
<b>2.2</b>	<b>Architecture diagram</b>
<b>3.1</b>	<b>Sprint Delivery Schedule</b>
<b>3.2</b>	<b>Sprint Report</b>
<b>4.1</b>	<b>Performance Metrics</b>

# **CHAPTER 1-INTRODUCTION**

## **1.1PROJECT OVERVIEW**

This project is based on expense tracking. This project aims to create an easy, faster and smooth cloud application .For better expense tracking we developed our project thatwill help the users a lot. Most of the people cannot track their expenses and income leading to facing money crisis, so this application can help people to track their expense day to day and make life stress free. Money is the most valuable portion of our daily life and without money we will not last one day on earth. So using the daily expense tracker application is important to lead a happy family. It helps the user to avoid unexpected expenses and bad financial situations. It will save time and provide a responsible lifestyle.

## **1.2PURPOSE**

Personal finance management is an important part of people's lives. However, everyone does not have the knowledge or time to manage their finances in a proper manner. And, even if a person has time and knowledge, they do not bother with tracking their expenses as they find it tedious and time-consuming. Now, you don't have to worry about managing your expenses, as you can get access to an expense tracker that will help in the active management of your finances.

Also known as expense manager and money manager, an expense tracker is a software or application that helps to keep an accurate record of your money inflow and outflow. Many people in India live on a fixed income, and they find that towards the end of the month they don't have sufficient money to meet their needs. While this problem can arise due to low salary, invariably it is due to poor money management skills.

People tend to overspend without realizing and this can prove to be disastrous. Using a daily expense manager can help you keep track of how much you spend every day and on what. At the end of the month, you will have a clear picture where your money is going. This is one of the best ways to get your expenses under control and bring some resemblance of order to your finances.

## CHAPTER 2 – LITERATURE SURVEY

### REFERENCES

**UPSingh, AKGupta, Dr.B.Balamurugan(2021)-Spending Tracker: A Smart Approach to Track Daily Expenses– Turkish Journal of Computer and Mathematics Education Vol.12No.6, 5095-5103**

In this paper, a Java GUI based application was proposed to assure that it will help its users to manage the cost of their daily expenditure. It will guide them and aware them of their daily expenses. The proposed design contained the basic modules for adding and viewing expenses, managing expense categories. Supports CRUD operations on expense data.

**Pros:**

- Category-wise management of expenses
- Daily,monthly,annual basis tracking
- User-friendly

**Cons:**

- Lack of visual analytics for expense data
- Lack of support for splitting group expenses
- Supports manual data monitoring only

**Prof Miriam Thomas, Lekshmi P, and Dr. Mahalekshmi T (2022) - Expense Tracker – International Journal of Advanced Research in Science, Communication and Technology (IJARSCT) Volume 9, Issue 4, September 2020 ISSN(Online) 2581-9429**

Daily Expense Tracker System is designed to keep a track of Income-Expense of an organisation on a day-to-day basis. This System divides the Income based on daily expenses. If exceed day's expense, system will calculate income and will provide new daily expense allowed amount. Daily expense tracking System will generate report at the end of month to show Income-Expense graph. And employees send reports to the manager for verification. Manager send final reports to administrator .Based on the final reports system predict the next month expense. It will help to manage over all expense and income.

**Pros:**

- Maintenance of expense data in the form of Excel sheets, CSV files, thereby avoiding entering individual expenses manually.
- Better visual analytics of data for various timelines.
- Supports handling for reimbursements
- Least squares regression, a statistical procedure, is used to predict the expense limits.

Cons:

- Suitable for organization scale, too complex for personal use.
- Expense prediction is not really necessary for small transactions made on personal use.
- Involves the participation of 3 roles–Admin, Manager, Employee

**Velmurugan A, Albert Mayan J, Niranjana P and Richard Francis (2020)  
– Expense Manager Application – Journal of Physics: Conference Series**

In this paper, a multi-purpose finance related android application intended to run on android devices is designed efficiently to give the best suggestions for finance planning and could be run on low-end devices with an application size as less as 10MB. It analyses and studies the pattern expenses in certain category or by distinct kinds of spending that can be used for studying market trends derived using some data mining techniques such as clustering, classification and association. Android Studio, Kotlin and Java, SQLite, Android OS, Figma Designing tools are used.

Pros:

- Tracks all daily transactions, suggesting efficient investment options
- Experience is hassle free and very handy
- Provided with authenticated finance

Cons:

- Application is yet modified to make it safe
- Modern mining methods are not used
- More visualization is required

**Velmurugan.R, Mrs.P.Usha (2021) – Expense Tracker Application-  
International Journal of Innovative Research in Technology (IJIRT) Volume  
7, Issue 10, March 2021 ISSN:2349-6002**

This is an android based application that allows user to maintain a computerized diary to track expenses on a day-to-day basis to stay on budget and know expenses that is represented via a graphical representation with special features of distributing expenses in different categories suitable for the user. Java, XML, MySQL is used. Filtering transaction views, view analytics and PDF reports are also included.

Pros:

- Has various components of updating and viewing users expenditure
- User can track his expenses by choosing a day and using various filtering options to study expenses
- Visualization using pie chart with percentage view shows graphical representation

Cons:

- Doesn't support upcoming android versions.
- If a particular data is deleted, it cannot be viewed again.
- Statistics about income and expense detail of user can be prepared.

**Saumya Dubey, Pragya Dubey, Rigved Rishabh Kumar, Aisha Khatoon-  
Student Expense Tracking Application-IJARIE-ISSN(O)-2395-4396 Vol-8  
Issue-2 2022**

This is an android application which is used to track the daily expense of a student. It is like a digital diary that keeps record of expenses done by a student. The application keeps track of money spent and the earnings both of the student on day-to-day basis. It also has the feature that it gives warning messages if we are exceeding on our expenses and hence, we can limit our expenses and avoid overspending. If you spend less money than the daily expense allowed amount, the money left after spending is added into user's savings.

Pros:

- This approach effectively keeps away from the manual figuring for trying not to ascertain the pay and cost each month.
- user-friendly

Cons:

- It does not provide any analytics.

**Namita Jagtap, Priyanka Joshi, Aditya Kamble (2019)- A Review on  
Budget Estimator Android Application-International Journal of  
Innovative Research in Technology (IJIRT) Volume6, Issue4, March  
2021 ISSN:2395-0056**

The system known as Budget Estimator is designed to manage the application user's daily expenses in a more efficient and manageable way. This project is about mobile application Expenses system with geo-location tracking, based on the location of the user, it using Google Places, to check, the available store in the area, provides a notification for offers purpose, In term of security design, this system may implement a login authentication such as OTP message to your mobile device, this function may bring more security confidence to user. To reduce manual calculations, we propose an application which is developed by android. This application allows users to maintain a digital automated diary.

Pros:

- In this paper, an algorithm was proposed to show offers in nearby places using geo-location tracking.
- This mobile application with 2-step verification method provides the security to the users.

Cons:

- Suitable for only Personal use.
- It does not provide any analytics.



**PROBLEMS IDENTIFIED:**

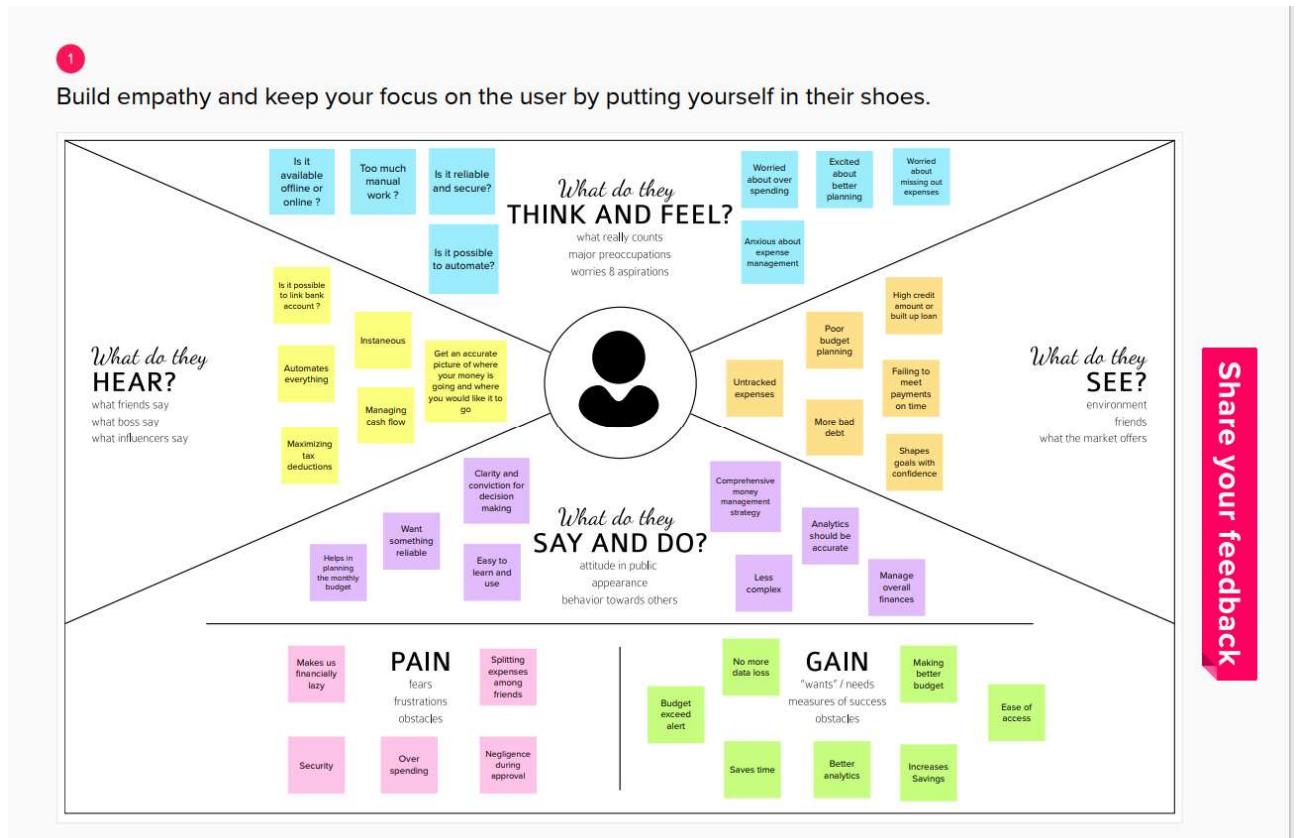
- Poor understanding of the spending patterns due to lack of analytics.
- No support for splitting group expenses, so need to manually calculate the expense share of each member.
- Regular report on the expense data is not provided.

**PROPOSED SOLUTIONS :**

- Various visualizations and analytics can be provided for better understanding of the spending patterns. Category-wise, daily, weekly, monthly, and annual analytics and report can be generated.
- The facility to split equally/unequally group expenses and notify group members the same.

## CHAPTER 3 – IDEATION AND PROPOSED SOLUTION

### 3.1 EMPATHY MAP CANVAS



### 3.2 IDEATION AND BRAINSTORMING


#### Brainstorm & Idea Prioritization Template:

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

# Step-1: Team Gathering, Collaboration and Select the Problem Statement

Template



## Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare

1 hour to collaborate

2-8 people recommended

➔

### Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A

#### Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B

#### Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.

C

#### Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

Open article

1

### Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

How might we organize expenses to provide better stats to the user securely?

Key rules of brainstorming

To run a smooth and productive session

Stay in topic.

Encourage wild ideas.

Defer judgment.

Listen to others.

# STEP-2:Brainstorm Idea Listing and Grouping

2

## Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

You can select a sticky note and hit the pencil (switch to sketch) icon to start drawing!

Ajitesh

Create a custom category

Encrypt transaction data

Split group expenses and notify members accordingly

Periodic and categorical analytics of expenses

Send reminders to members and automatically match them with the right invoices

Deekshith

Categorize the expenses

Keep a captured record of receipts

Predict next purchase according to spending behaviour

Voice assistant

Automates the payments by scanning the receipt and categorizing the

Eswaramoorthy

Link to bank and UPI accounts

Keep track of different payment methodologies like credit and debit card, net banking, mobile wallets and bank transfers

Display all transactions and provide filters

Timely tips on investment and expenses

A chat bot that covers all queries and raises tickets for errors and bugs not provided by the app

Hemanth

Read SMS messages of transactions and record them

Insights about budget vs actual spent

Provide third party integration with app that provide third party group interest setting

CRUD operations on expense data

Multi-lingual support

Vishnupriya

A budget exceed alert on mail or SMS

Automate calculations for immediate reimbursement and settlements

Generate and send reports to give detailed insights

Provide a good UI/UX design that makes it easier for any user to adopt quickly

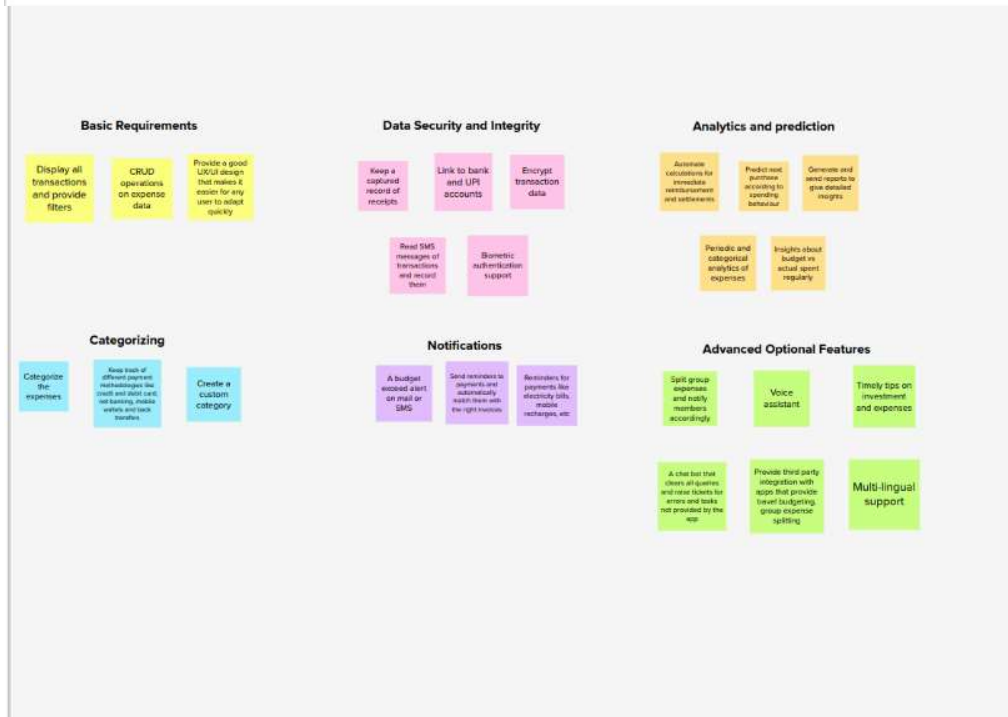
Biometric authentication support

3

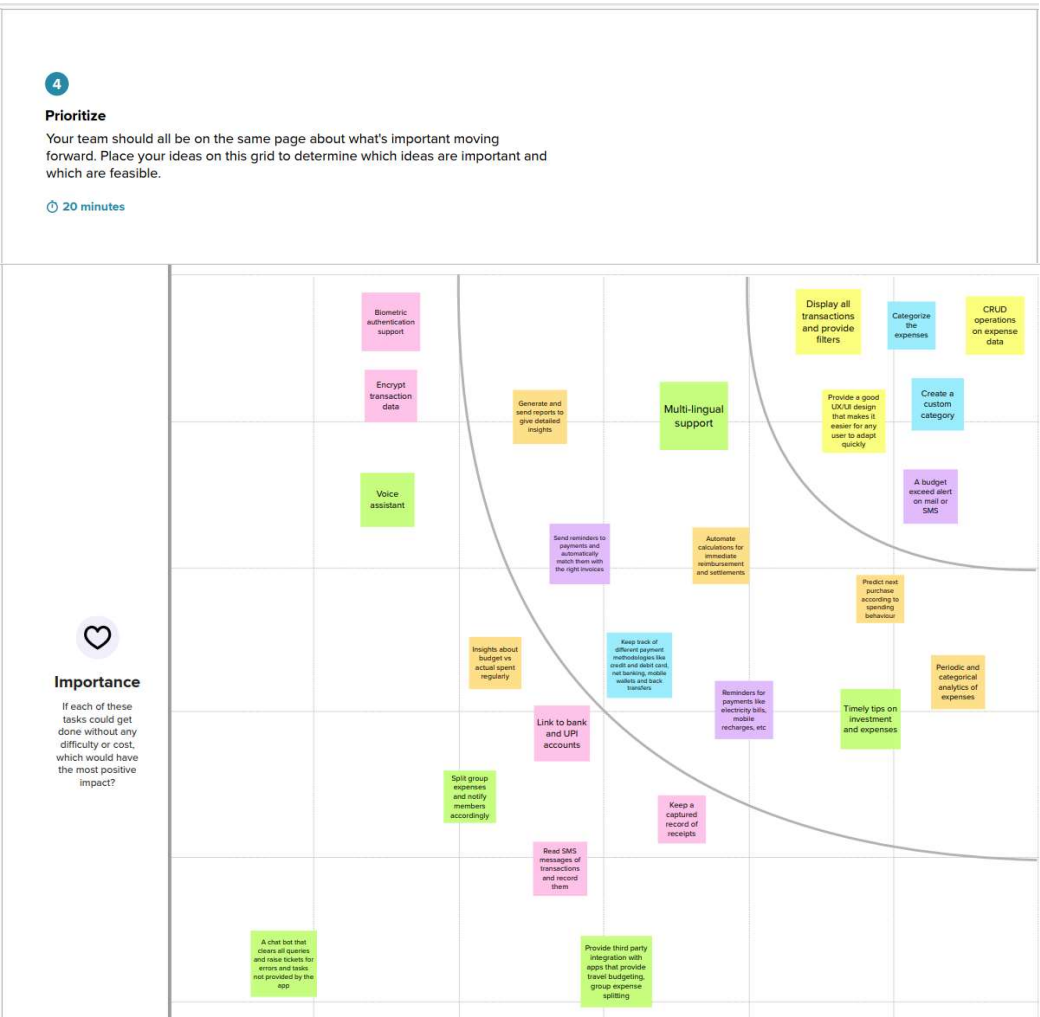
### Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

🕒 20 minutes



# Step-3:Idea Prioritization



### 3.3 PROPOSED SOLUTION

#### Proposed Solution Template:

Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The existing manual expense tracking system is tiresome, fault intolerant and not reliable. It is also hard to analyse our spending pattern.
2.	Idea / Solution description	Personal expense tracker system is designed for both salaried and non-salaried personnel for keeping track of their day-to-day expenses effectively through a computerised system, which eliminates the tedious manual paperworks. The users are asked to enter their monthly budget. When the user add his expenses, it is reflected in his wallet balance. The user also receives an email alert if his expenses exceed the monthly budget. Also the tracking system provides various visual analytics and generate report on the user's spending data.
3.	Novelty / Uniqueness	The user gets notified when their expenses exceed the monthly limit through email. Generation of visual analytics and monthly report on expenses are included.
4.	Social Impact / Customer Satisfaction	This application makes user life easy as it reduces lot of manual paper work. It also generate reports of their spending and notify users if they have exceeded their budget. This application creates self awareness on the spending patterns of the user. It helps user to be financially responsible.
5.	Business Model (Revenue Model)	This application is provided for free of cost. But It will have some advertisements. In premium version there are no advertisements and
		contains additional features like splitting group expenses, expense prediction, etc
6.	Scalability of the Solution	The application can handle large number of users and data with high performance and security . This application can handle expense data of varying scale. Easily available in multiple devices.

### 3.4 PROPOSED SOLUTION FIT

Project Title: Personal Expense Tracker

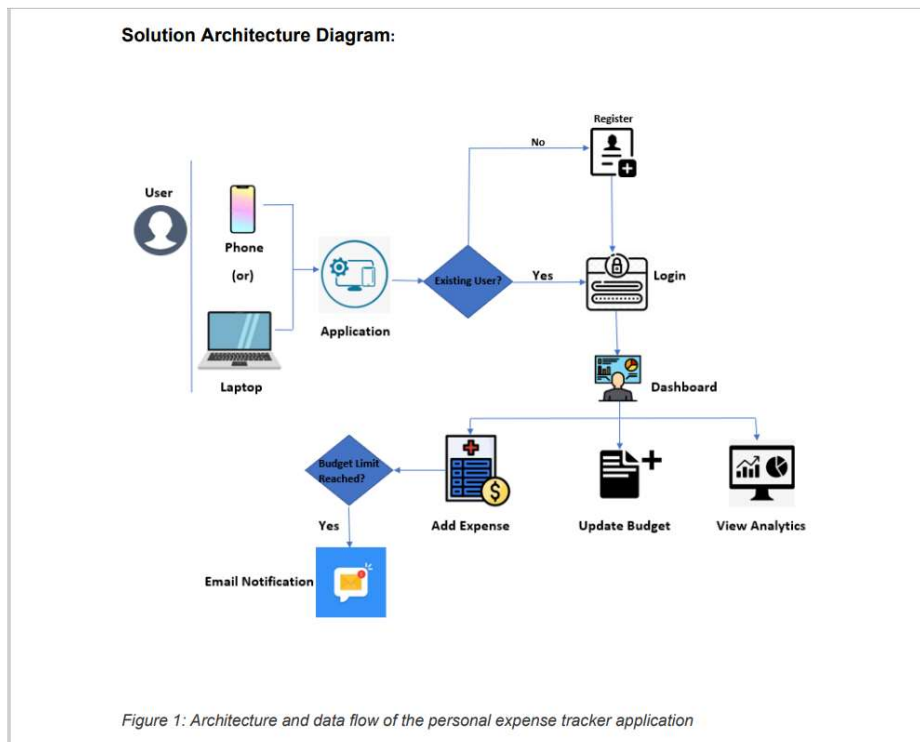
Project Design Phase-I - Solution Fit Template

Team ID: PNT2022TMD35249

Define CS, fit into CC	<b>1. CUSTOMER SEGMENT(S)</b> <ul style="list-style-type: none"> <li>Working Individuals</li> <li>Students</li> <li>Budget conscious consumers</li> </ul>	<b>6. CUSTOMER CONSTRAINTS</b> <ul style="list-style-type: none"> <li>Internet access</li> <li>Device (Smartphones) to access the application</li> <li>Awareness of the functionality</li> <li>Initially provided with free cost for existing applications</li> <li>Data privacy and trust</li> <li>Not all transactions are made online</li> </ul>	<b>5. AVAILABLE SOLUTIONS</b> <ul style="list-style-type: none"> <li>Expense Diary or Excel sheet</li> </ul> <p>PROS: We make a note daily so we are constantly aware, privacy and security</p> <p>CONS: Takes a lot of time, we might miss some transactions and can be wrong, Inconvenient</p>	Explore AS, differentiate
	<b>2. JOBS-TO-BE-DONE / PROBLEMS</b> <ul style="list-style-type: none"> <li>To keep track of money lent or borrowed</li> <li>To keep track of daily transactions</li> <li>Alert when a threshold limit is reached</li> </ul>	<b>9. PROBLEM ROOT CAUSE</b> <ul style="list-style-type: none"> <li>Procastination and reckless spending</li> <li>Indecisive about finances</li> <li>Difficult to maintain a note of daily spendings (Traditional methods like diary)</li> <li>Shopping to fill the inner void and to make feel better about ourself</li> <li>Attractive offers that attract us to buy more than needed</li> </ul>	<b>7. BEHAVIOUR</b> <ul style="list-style-type: none"> <li>Browse through the internet to find solutions</li> <li>Consult friends and relatives for help</li> <li>Completely reduce spendings or spend all of the savings</li> <li>Make use of online tools to interpret monthly expense patterns</li> </ul>	

<p><b>3. TRIGGERS</b></p> <ul style="list-style-type: none"> <li>Excessive spending</li> <li>No money in case of emergency</li> <li>When they are referred by someone who has experienced it</li> <li>When their expenses go overboard, there is a sense of panic to get things right</li> </ul>	<p><b>10. YOUR SOLUTION</b></p> <ul style="list-style-type: none"> <li>Create an application to manage the expenses of an individual in an efficient and manageable manner, as compared to traditional methods</li> <li>Providing timely reminders, helping customers to prioritize their spends, allocation of budget daily</li> </ul>	<p><b>8. CHANNELS of BEHAVIOUR</b></p> <ol style="list-style-type: none"> <li>ONLINE: <ul style="list-style-type: none"> <li>Find more information on the product by surfing the internet</li> <li>Contact the ones who have already been using the service</li> <li>Maintain excel sheets and use visualizing tools</li> <li>Read reviews on the internet, compare them with different services</li> </ul> </li> <li>OFFLINE: <ul style="list-style-type: none"> <li>Visit experience centers to know more about the product</li> <li>Negotiation with the vendor</li> <li>Consult with friends and relatives for suggestions</li> <li>Maintain an expense diary</li> </ul> </li> </ol>
<p><b>4. EMOTIONS: BEFORE / AFTER</b></p> <ul style="list-style-type: none"> <li>BEFORE: Lost, Misguided, Disorganized, Manipulated, Anxiety, Confused, Fear</li> <li>AFTER: Confident, Organized, Happy, Content, Composed, Calm</li> </ul>		

## 3.5 SOLUTION ARCHITECTURE



## CHAPTER 4 – REQUIREMENT ANALYSIS

### 4.1 FUNCTIONAL REQUIREMENTS

FRNo.	Functional Requirement(Epic)	Sub Requirement (Story/Sub-Task)
FR-1	User Registration	Registration through Form
FR-2	Login	Enter the username and password
FR-3	Calendar	Personal expense tracker application shall allow user to add the data to their expenses.
FR-4	ExpenseTracker	This application should graphically represent the expense in the form of report.
FR-5	Report generation	Graphical representation of report.
FR-6	Category	This application shall allow users to add categories of their expenses.

### 4.2 NON-FUNCTIONAL REQUIREMENTS

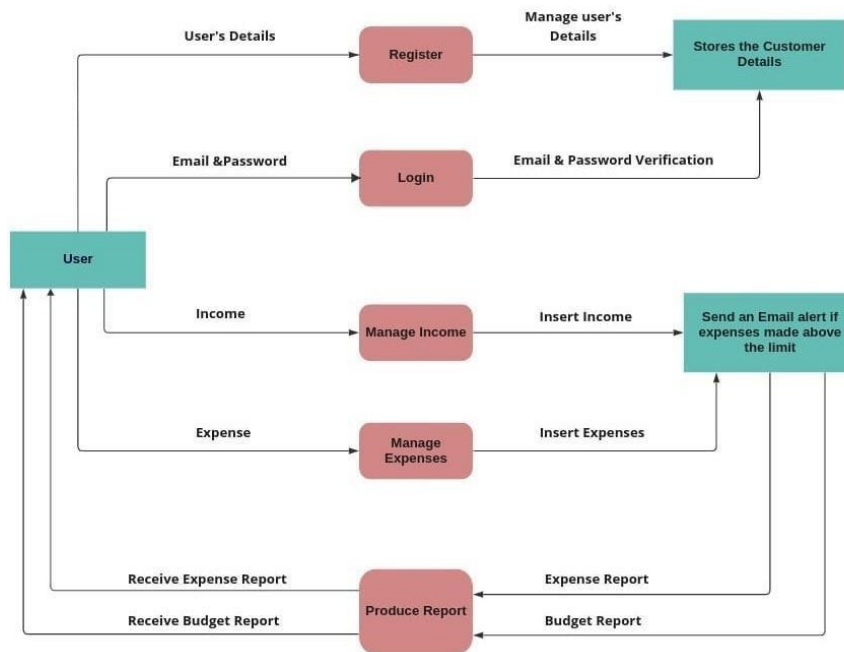
FRNo.	Non-Functional Requirement	Description
NFR-1	Usability	Helps to keep an accurate record of your money inflow and outflow.
NFR-2	Security	Budget tracking apps are considered very safe from cyber criminals.
NFR-3	Reliability	Each data record is stored on a well built efficient database schema. There is no risk of data loss
NFR-4	Performance	The types of expense are categories along with an option. Throughput of the system is increased due to lightweight data base support.
NFR-5	Availability	It is available all the time.
NFR-6	Scalability	The ability to appropriately handle increasing demands.



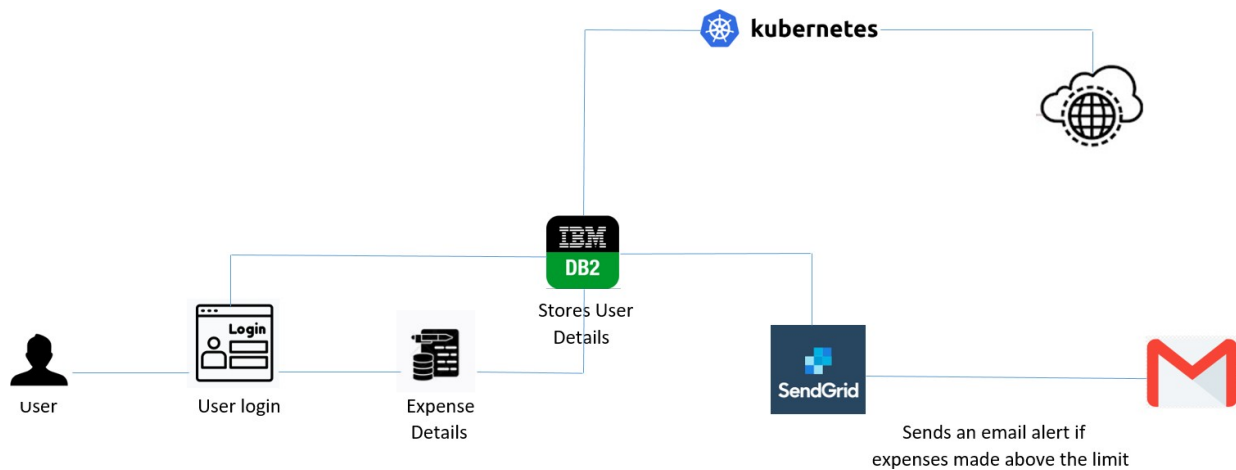
## CHAPTER 5 – PROJECT DESIGN

### 5.1 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



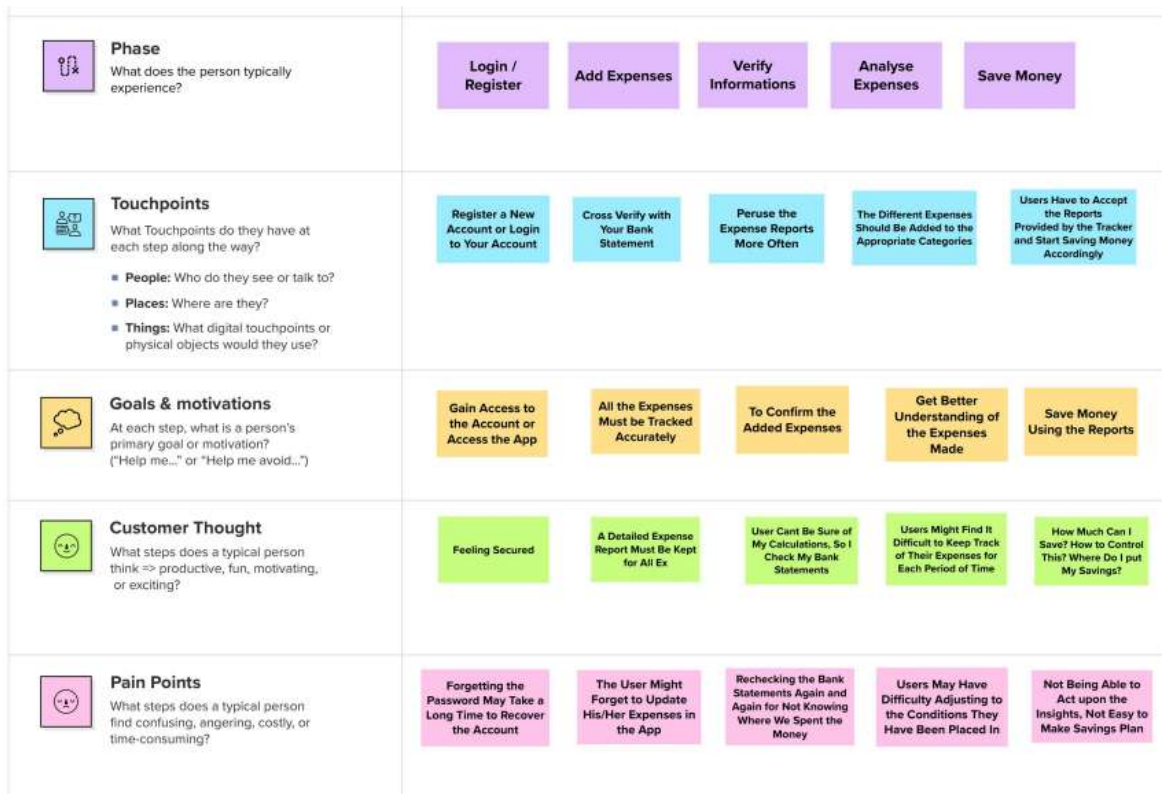
### 5.2 TECHNICAL ARCHITECTURE



## 5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account /dashboard	High	
	Login	USN-2	As a user, I can log into the application by entering email & password	I can access the application	High	
	Dashboard	USN-3	As a user I can enter my income and expenditure details.	I can view my daily expenses	High	
Customer Care Executive		USN-4	As a customer care executive, I can solve the log in issues and other issues of the application.	I can provide support or solution at any time 24*7	Medium	
Administrator	Application	USN-5	As an administrator I can upgrade or update the application.	I can fix the bug which arises for the customers and users of the application	Medium	

## 5.4 CUSTOMER JOURNEY MAP



## CHAPTER 6 – PROJECT PLANNING AND SCHEDULING

### 6.1 SPRINT PLANNING AND ESTIMATION

TITLE	DESCRIPTION	DATE
<b>Literature Survey &amp; Information Gathering</b>	A literature review is a comprehensive summary of previous research on a topic. The collection / gathering of relevant information based on our project use case and by referring the existing solutions etc.	13 OCTOBER 2022
<b>Prepare Empathy Map</b>	Empathy map canvas is prepared to identify the customer or user's Pains & Gains and their feelings and thinking and list of Problem statement is prepared.	13 OCTOBER 2022
<b>Ideation</b>	In ideation , Ideas are listed in 3 steps like the first step is problem statement, second step is brainstorming, idea listing and grouping and third step is Idea prioritization based on the feasibility & importance are prepared and submitted for review	13 OCTOBER 2022
<b>Proposed Solution</b>	Proposed solution includes the problem statement, idea, novelty, Customer satisfaction, business model, social impact, scalability of solutions are prepared	13 OCTOBER 2022

<b>Problem Solution Fit</b>	Problem-solution fit document includes customer segment, problems, triggers, Emotions before and after, available solutions, Customer constraint, Behavior, problem root cause, your solution these things are prepared.	18 OCTOBER 2022
<b>Solution Architecture</b>	Prepare solution architecture document	17 OCTOBER 2022

<b>Customer Journey</b>	Customer journey helpful to identify and understand the user interactions, goals and opportunities, positive and negative moments & experiences with the application (entry to exit).	15 OCTOBER 2022
<b>Functional Requirement</b>	Functional requirement document having the functionalities and non-functionalities of our project	16 OCTOBER 2022
<b>Data Flow Diagrams</b>	Data flow diagrams show the flow of our project and it has been done and submitted for review.	1 NOVEMBER 2022
<b>Technology Architecture</b>	Technology Architecture has been done and submitted for review.	18 OCTOBER 2022
<b>Prepare Milestone &amp; Activity List</b>	Prepared the milestones & activity list of the project.	3 NOVEMBER 2022

<b>Project Development - Delivery of Sprint-1, 2, 3 &amp; 4</b>	Develop & submit the developed code by testing it.	24 OCTOBER 2022 - 19 NOVEMBER 2022
---	--	--

## 6.2 SPRINT DELIVERY SCHEDULE

### Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	3	High	Vishnu priya.N
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	3	High	Eshwarmoorthy.K
Sprint-1	Login	USN-3	As a user, I can log into the application by entering email & password	5	High	Vishnupriya.N
Sprint-1	Dashboard & Logout	USN-4	As a user, once I logged in I can access all the features of the web app and Logout once I completed all the work.	5	High	Hemanth.D
Sprint-1		USN-5	Once logged In, Keep me logged for few hours to avoid repeated login if the page is refreshed	4	Medium	Hemanth.D
Sprint-2	Expense	USN-6	Add total income for the month and Allow for edit option	6	High	Eshwarmoorthy.K
Sprint-2		USN-7	Split the total income based on usage like entertainment, food, shopping etc.	2	Low	Vishnupriya.N
Sprint-2		USN-8	Add the day to day expense.	6	High	Ajitesh.M
Sprint-2		USN-9	Display the user added expense	6	High	Deekshith.M
Sprint-3		USN-10	Filter the expense data based on criteria	6	Medium	Split Between 5
Sprint-3	Charts	USN-11	As a user I can display it in graphs	4	Low	Vishnupriya.N
Sprint-3	Alerts	USN-12	As a user I create custom alert for the balance	10	High	Deekshith.M
Sprint-4	Deployment	USN-13	As a user I should able to access it anywhere in the net	20	High	Ajitesh.M

### Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	3 Days	5 Nov2022	7 Nov 2022	20	7 Nov 2022
Sprint-2	20	3 Days	8 Nov 2022	10 Nov 2022	20	8 Nov 2022
Sprint-3	20	4 Days	11 Nov 2022	14 Nov 2022	20	14 Nov 2022
Sprint-4	20	6 Days	15 Nov 2022	20 Nov 2022	20	25 Nov 2022

## Velocity:

We have a 6-day sprint duration, and the velocity of the team is 20 (points per sprint). Calculating the team's average velocity (AV).

$$AV = \text{sprintduration} = 20 / 6 = 3.33 \text{ velocity}$$

## CHAPTER 7 – CODING AND SOLUTIONING

Code:

```
from flask import Flask, render_template, request, redirect, session
import re
import ibm_db
import dateutil.parser
from sendemail import *

app = Flask(__name__)

app.secret_key = 'your secret key'
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=2f3279a5-73d1-4859-88f0-
a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=30756;SECURITY=SSL
;SSLServerCertificate=DigiCertGlobalRootCA.crt;PROTOCOL=TCP;UID=bkq23793;PWD=6sWR
xHNJMuQyhfgs;", "", "")

@app.route("/home")
def home():
    return render_template("homepage.html")

@app.route("/")
def add():
    return render_template("home.html")

@app.route("/signup")
def signup():
    return render_template("signup.html")

@app.route('/register', methods=['GET', 'POST'])
def register():
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']

        queryfind = "SELECT * FROM register WHERE username = '{}'.format(username)
        out = ibm_db.exec_immediate(conn,queryfind)
        account = ibm_db.fetch_assoc(out)
        print(account)
        if account:
            msg = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
        else:
            query3 = "INSERT INTO register VALUES ('{}', '{}',
'{}'.format(username,email,password)
            ibm_db.exec_immediate(conn,query3)
            msg = 'You have successfully registered !'
            return render_template('signup.html', msg = msg)

@app.route("/signin")
def signin():
    return render_template("login.html")

@app.route('/login',methods=['GET', 'POST'])
def login():
    global userid
```

```

msg = ''

if request.method == 'POST' :
    username = request.form['username']
    password = request.form['password']
    query = "SELECT * FROM register WHERE username = '{}' AND password = '{}'".format(username,password)
    out = ibm_db.exec_immediate(conn,query)
    account = ibm_db.fetch_assoc(out)
    print (account)

    if account:
        session['loggedin'] = True
        session['id'] = account['EMAIL']
        userid= account['EMAIL']
        session['username'] = account['USERNAME']

        return redirect('/home')
    else:
        msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)

@app.route("/add")
def adding():
    return render_template('add.html')

@app.route('/addexpense',methods=['GET', 'POST'])
def addexpense():

    date = request.form['date']
    print(date)
    date = dateutil.parser.parse(date).strftime('%Y-%m-%d')
    expensename = request.form['expensename']
    amount = request.form['amount']
    paymode = request.form['paymode']
    category = request.form['category']

    query3 = "INSERT INTO expenses(email,date,expensename,amount,paymode,category)
VALUES ('{}', '{}', '{}', '{}', '{}', '{}')".format(session['id'],date,
expensename, amount, paymode, category)
    ibm_db.exec_immediate(conn,query3)
    print(date + " " + expensename + " " + amount + " " + paymode + " " + category)

    return redirect("/display")

@app.route("/display")
def display():
    print(session["username"],session['id'])
    queryfind = "SELECT * FROM expenses WHERE email = '{}' ORDER BY date
DESC".format(str(session['id']))
    out = ibm_db.exec_immediate(conn,queryfind)
    account = ibm_db.fetch_tuple(out)
    list1 = []
    list2 = []
    while account!=False:
        list2.append(account[0])
        list2.append(account[1])
        list2.append(account[2])
        list2.append(account[3])
        list2.append(account[4])
        list2.append(account[5])
        list2.append(account[6])
        list1.append(list2.copy())
        list2.clear()
        account = ibm_db.fetch_tuple(out)

```

```

print(account)

return render_template('display.html' ,expense = list1)

@app.route('/delete/<string:id>', methods = ['POST', 'GET' ])
def delete(id):
    query3 = "DELETE FROM expenses WHERE id = '{}'.format(id)
    ibm_db.exec_immediate(conn,query3)
    print('deleted successfully')
    return redirect("/display")

@app.route('/edit/<id>', methods = ['POST', 'GET' ])
def edit(id):
    queryfind = "SELECT * FROM expenses WHERE id = '{}'.format(id)
    out = ibm_db.exec_immediate(conn,queryfind)
    row = ibm_db.fetch_tuple(out)

    print(row[0])
    return render_template('edit.html', expenses = row)

@app.route('/update/<id>', methods = ['POST'])
def update(id):
    if request.method == 'POST' :
        date = request.form['date']
        print(date)
        date = dateutil.parser.parse(date).strftime('%Y-%m-%d')
        expensename = request.form['expensename']
        amount = request.form['amount']
        paymode = request.form['paymode']
        category = request.form['category']
        query = "UPDATE expenses SET date = '{}', expensename = '{}', amount = '{}',
paymode = '{}', category = '{} WHERE id = '{}'.format(date, expensename, amount,
str(paymode), str(category),id)
        ibm_db.exec_immediate(conn,query)
        print('successfully updated')
        return redirect("/display")

@app.route("/limit" )
def limit():
    return redirect('/limitn')

@app.route("/limitnum" , methods = ['POST' ])
def limitnum():
    if request.method == "POST":
        number= request.form['number']
        query = "INSERT INTO limits(email,limitamt) VALUES ('{}', '{}')
".format(session['id'], number)
        ibm_db.exec_immediate(conn,query)
        return redirect('/limitn')

@app.route("/limitn")
def limitn():
    query = "SELECT limitamt FROM limits WHERE email = '{} ORDER BY id DESC LIMIT
1".format(session['id'])
    out = ibm_db.exec_immediate(conn,query)
    x = ibm_db.fetch_tuple(out)
    s = x

    return render_template("limit.html" , y= s)

@app.route("/today")
def today():
    query = "SELECT DATE(date) , amount FROM expenses WHERE email = '{}' AND
DATE(date) = DATE(NOW()) ".format(str(session['id']))

```



```

out = ibm_db.exec_immediate(conn,query)
account = ibm_db.fetch_tuple(out)
texpanse = []
list2 = []
while account!=False:
    list2.append(account[0])
    list2.append(account[1])
    texpanse.append(list2.copy())
    list2.clear()
    account = ibm_db.fetch_tuple(out)

query2 = "SELECT * FROM expenses WHERE email = '{}' AND DATE(date) =
DATE(NOW()) ORDER BY date DESC".format(str(session['id']))
out = ibm_db.exec_immediate(conn,query2)
account = ibm_db.fetch_tuple(out)
expense = []
list2 = []
while account!=False:
    list2.append(account[0])
    list2.append(account[1])
    list2.append(account[2])
    list2.append(account[3])
    list2.append(account[4])
    list2.append(account[5])
    list2.append(account[6])
    expense.append(list2.copy())
    list2.clear()
    account = ibm_db.fetch_tuple(out)
total=0
t_food=0
t_entertainment=0
t_business=0
t_rent=0
t_EMI=0
t_other=0

for x in expense:
    total += x[4]
    if x[6] == "food":
        t_food += x[4]

    elif x[6] == "entertainment":
        t_entertainment += x[4]

    elif x[6] == "business":
        t_business += x[4]
    elif x[6] == "rent":
        t_rent += x[4]

    elif x[6] == "EMI":
        t_EMI += x[4]

    elif x[6] == "other":
        t_other += x[4]

print(total)

print(t_food)
print(t_entertainment)
print(t_business)
print(t_rent)
print(t_EMI)
print(t_other)

return render_template("today.html", texpanse = texpanse, expense = expense,
total = total ,
                        t_food = t_food,t_entertainment = t_entertainment,

```

```

        t_business = t_business, t_rent = t_rent,
        t_EMI = t_EMI, t_other = t_other )

@app.route("/month")
def month():
    query = "SELECT DATE(date) , SUM(amount) FROM expenses WHERE email = '{}'
    AND MONTH(DATE(date)) = MONTH(NOW()) GROUP BY DATE(date) ORDER BY
    DATE(date)".format(str(session['id']))
    out = ibm_db.exec_immediate(conn,query)
    account = ibm_db.fetch_tuple(out)
    texpense = []
    list2 = []
    while account!=False:
        list2.append(account[0])
        list2.append(account[1])
        texpense.append(list2.copy())
        list2.clear()
        account = ibm_db.fetch_tuple(out)

    #cursor = mysql.connection.cursor()
    #cursor.execute('SELECT * FROM expenses WHERE userid = % s AND DATE(date) =
    DATE(NOW()) AND date ORDER BY `expenses`.`date` DESC',(str(session['id'])))
    query2 = "SELECT * FROM expenses WHERE email = '{}' AND MONTH(DATE(date)) =
    MONTH(NOW()) ORDER BY date DESC".format(str(session['id']))
    #expense = cursor.fetchall()
    #print(expense)
    out = ibm_db.exec_immediate(conn,query2)
    account = ibm_db.fetch_tuple(out)
    expense = []
    list2 = []
    while account!=False:
        list2.append(account[0])
        list2.append(account[1])
        list2.append(account[2])
        list2.append(account[3])
        list2.append(account[4])
        list2.append(account[5])
        list2.append(account[6])
        expense.append(list2.copy())
        list2.clear()
        account = ibm_db.fetch_tuple(out)
    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0

    for x in expense:
        total += x[4]
        if x[6] == "food":
            t_food += x[4]

        elif x[6] == "entertainment":
            t_entertainment += x[4]

        elif x[6] == "business":
            t_business += x[4]
        elif x[6] == "rent":
            t_rent += x[4]

        elif x[6] == "EMI":
            t_EMI += x[4]

        elif x[6] == "other":
            t_other += x[4]

```

```

        query = "SELECT limitamt FROM limits WHERE email = '{} ' ORDER BY id DESC LIMIT
1".format(session['id'])
        #x= cursor.fetchone()
        out = ibm_db.exec_immediate(conn,query)
        x = ibm_db.fetch_tuple(out)
        s = x[0]
        if(total>s):
            sendmail("You have exceeded your monthly limit! Please visit the history
section to know more about your expenses.", session['id'])
            print(total)

            print(t_food)
            print(t_entertainment)
            print(t_business)
            print(t_rent)
            print(t_EMI)
            print(t_other)

        return render template("today.html", texpanse = texpanse, expense = expense,
total = total ,
                                t_food = t_food,t_entertainment = t_entertainment,
                                t_business = t_business, t_rent = t_rent,
                                t_EMI = t_EMI, t_other = t_other )

@app.route("/year")
def year():
    query = "SELECT MONTH(date) , SUM(amount) FROM expenses WHERE email = '{} '
AND YEAR( DATE(date)) = YEAR(NOW()) GROUP BY MONTH(date) ORDER BY
MONTH(date)".format(str(session['id']))
    out = ibm_db.exec_immediate(conn,query)
    account = ibm_db.fetch_tuple(out)
    texpanse = []
    list2 = []
    while account!=False:
        list2.append(account[0])
        list2.append(account[1])
        texpanse.append(list2.copy())
        list2.clear()
        account = ibm_db.fetch_tuple(out)

    query2 = "SELECT * FROM expenses WHERE email = '{} ' AND YEAR( DATE(date)) =
YEAR(NOW()) ORDER BY date DESC".format(str(session['id']))

    out = ibm_db.exec_immediate(conn,query2)
    account = ibm_db.fetch_tuple(out)
    expense = []
    list2 = []
    while account!=False:
        list2.append(account[0])
        list2.append(account[1])
        list2.append(account[2])
        list2.append(account[3])
        list2.append(account[4])
        list2.append(account[5])
        list2.append(account[6])
        expense.append(list2.copy())
        list2.clear()
        account = ibm_db.fetch_tuple(out)
    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0

```

```

t_other=0

for x in expense:
    total += x[4]
    if x[6] == "food":
        t_food += x[4]

    elif x[6] == "entertainment":
        t_entertainment += x[4]

    elif x[6] == "business":
        t_business += x[4]
    elif x[6] == "rent":
        t_rent += x[4]

    elif x[6] == "EMI":
        t_EMI += x[4]

    elif x[6] == "other":
        t_other += x[4]

print(total)

print(t_food)
print(t_entertainment)
print(t_business)
print(t_rent)
print(t_EMI)
print(t_other)

return render_template("today.html", texpense = texpense, expense = expense,
total = total ,
                                t_food = t_food,t_entertainment = t_entertainment,
                                t_business = t_business, t_rent = t_rent,
                                t_EMI = t_EMI, t_other = t_other )

@app.route('/logout')

def logout():
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('username', None)
    return render_template('home.html')

if __name__ == "__main__":
    app.run(debug=True)

```

## Sending mail:

```

import smtplib
import sendgrid
import os
from sendgrid.helpers.mail import Mail, Email, To, Content

SUBJECT = "expense tracker"
s = smtplib.SMTP('smtp.gmail.com', 587)

def sendmail(TEXT,email):
    print("sorry we cant process your candidature")
    s = smtplib.SMTP('smtp.gmail.com', 587)
    s.starttls()

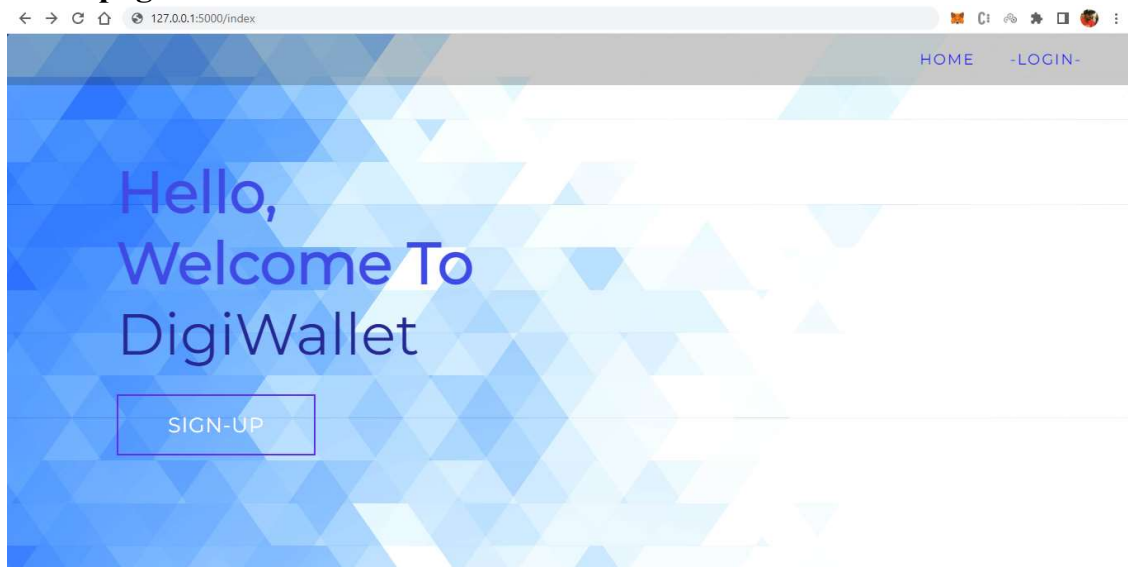
```

```
s.login("tproduct8080@gmail.com", "lxixbmpnexbkiemh")
message = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)
s.sendmail("il.shridhartp24@gmail.com", email, message)
s.quit()
def sendgridmail(user,TEXT):
    from_email = Email("shridhartp24@gmail.com")
    to_email = To(user)
    subject = "Sending with SendGrid is Fun"
    content = Content("text/plain",TEXT)
    mail = Mail(from_email, to_email, subject, content)

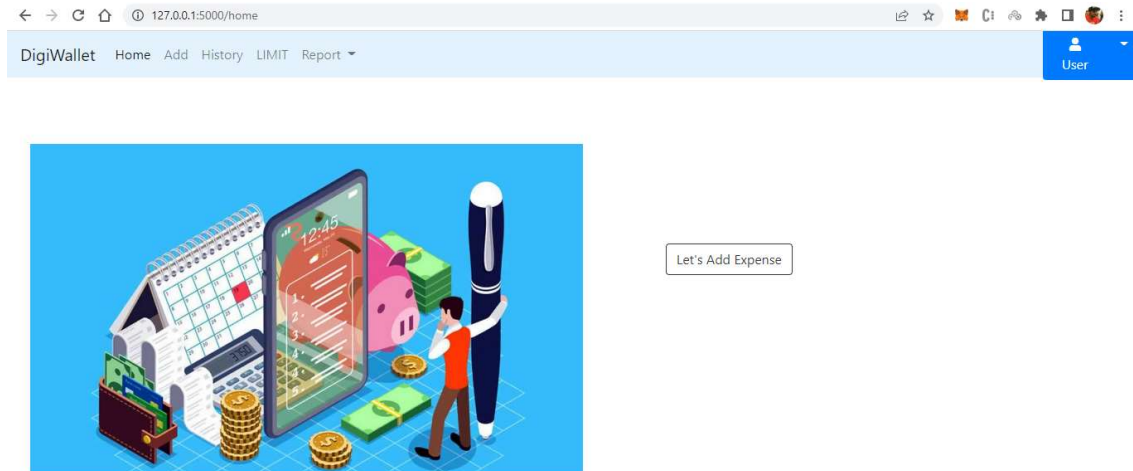
    # Get a JSON-ready representation of the Mail object
    mail_json = mail.get()
    # Send an HTTP POST request to /mail/send
    response = sg.client.mail.send.post(request_body=mail_json)
    print(response.status_code)
    print(response.headers)
```

## Output:

### Index page



## Home page after login



## Expense history

The screenshot shows the DigiWallet expense history page. The address bar displays '127.0.0.1:5000/display'. The navigation bar is identical to the home page. The main content area is titled 'EXPENSES' and displays a table of transactions.

Date	Description	Amount	Category	Action
2022-11-19	Netflix	₹ 199	entertainment	Edit Delete
2022-11-18	Recharge	₹ 1000	onlinebanking	Edit Delete

## Adding expense

← → ↻ 🏠 127.0.0.1:5000/add

DigiWallet Home Add History LIMIT Report User

### Add Expense

Date  
11/19/2022 09:52 PM


Expense name  
Netflix

Expense Amount  
199

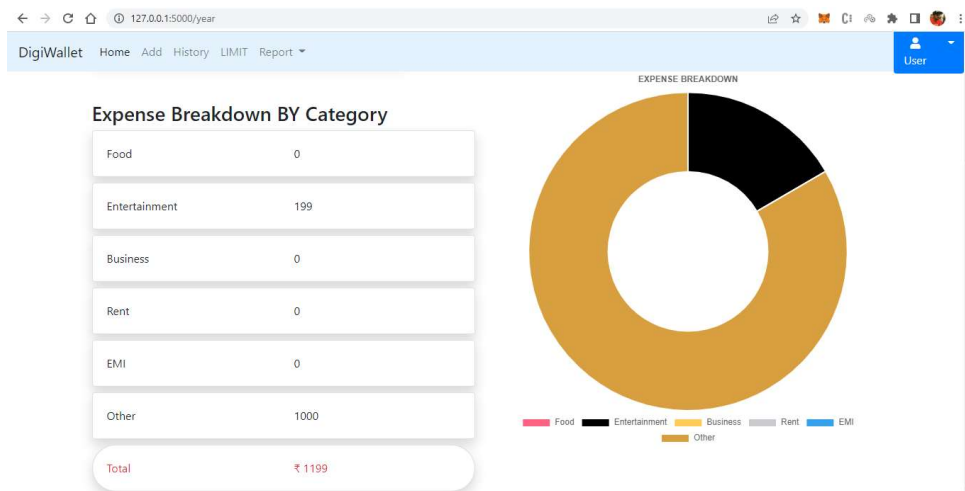
epayment

Entertainment

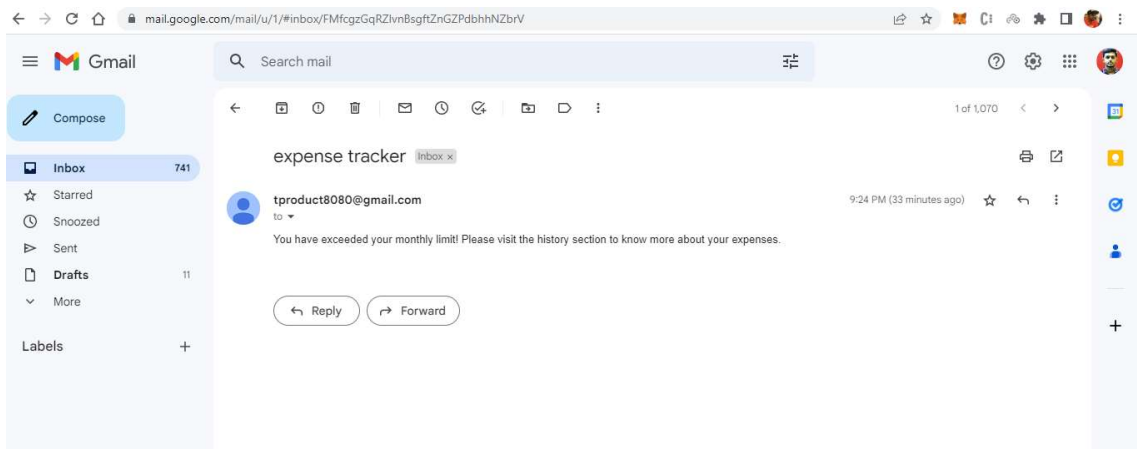
Add



## Expense Report



# Email alerting user about monthly limit





## CHAPTER 8 - TESTING

### 8.1 TEST CASES

1. Login with incorrect credentials
2. Signup with an already existing mail ID
3. Signup with wrong form data entered
4. Entering homepage with logged out session
5. Delete expense trigger change in graph
6. Add expense without choosing category

### 8.2 USER ACCEPTANCE TESTING

S. No	Test Case id	Feature Type	Test description	Input test Data	Actual output	Expected output	Remarks
1	TC – RG 01	Functional	Register for application by entering my name, email, password, monthly limit	User1@gmail.com ***** * 5000	Registration successful	Registration successful	pass
2	TC – SI 01	Functional	Log into the application by entering Email & password	User1@gmail.com *****	Login successful	Login successful	pass
3	TC – ST 01	UI	View my entire expenses throughout a particular period of time		Expenses are displayed for particular time	Expenses are displayed for particular time	pass

4	TC – DB 01	UI	Display graph in dashboard		Graph is displayed	Graph is displayed	pass
5	TC – ST 02	Functional	Generate reports based on my previous expenditures		Reports generated in graphical form	Reports generated in graphical form	pass
6	TC – SI 02	Functional	Can logout		Go to sign page	Sign in page displayed	pass
7	TC – ST 03	Functional	Create expense	14-11-2022 100 Food	Expenses created	Expenses created	pass
8	TC – ST 04	Functional	Can edit ,delete, update expense		Expenses updated	Updated of expenses	pass
9	TC – ST 05	UI	Can view Credit and debit expenses separately.		Expenses are listed separately	Expenses are listed separately	pass
10	TC – ST 06	UI	Aware of the expense that I spend the most on		Expenses are listed for particular category	Expenses are listed for particular category	pass
11	TC – PG 01	Functional	Able to update my set monthly limit		Monthly limit updated	Monthly limit updated	pass
12	TC – PG 01	UI	Able to View my profile		Profile details displayed	Profile details displayed	pass

## **CHAPTER 9 – RESULTS**

### **9.1 PERFORMANCE METRICS**

- i. Tracking income and expenses: Monitoring the income and tracking all expenditures (through bank accounts, mobile wallets, and credit & debit cards).
- ii. Transaction Receipts: Capture and organize your payment receipts to keep track of your expenditure.
- iii. Organizing Taxes: Import your documents to the expense tracking app, and it will streamline your income and expenses under the appropriate tax categories.
- iv. Payments & Invoices: Accept and pay from credit cards, debit cards, netbanking, mobile wallets, and bank transfers, and track the status of your invoices and bills in the mobile app itself. Also, the tracking apps sends reminders for payments and automatically matches the payments with invoices.
- v. Reports: The expense tracking app generates and sends reports to give a detailed insight about profits, losses, budgets, income, balance, sheets, etc.
- vi. Ecommerce integration: Integrate your expense tracking app with your eCommerce store and track your sales through payments received via multiple payment methods.
- vii. Vendors and Contractors: Manage and track all the payments to the vendors and contractors added to the mobile app.

## CHAPTER 10 – ADVANTAGES AND DISADVANTAGES

1. **Achieve your business goals** with a tailored mobile app that perfectly fits your business.
2. **Scale-up** at the pace your business is growing.
3. Deliver an **outstanding** customer experience through additional control over the app.
4. Control the **security** of your business and customer data
5. Open **direct marketing channels** with no extra costs with methods such as push notifications.
6. **Boost the productivity** of all the processes within the organization.
7. Increase **efficiency** and **customer satisfaction** with an app aligned to their needs.
8. **Seamlessly integrate** with existing infrastructure.
9. Ability to provide **valuable insights**.
10. Optimize sales processes to generate **more revenue** through enhanced data collection.

## CHAPTER 11 – CONCLUSION

From this project, we are able to manage and keep tracking the daily expenses as well as income. While making this project, we gained a lot of experience of working as a team. We discovered various predicted and unpredicted problems and we enjoyed a lot solving them as a team. We adopted things like video tutorials, text tutorials, internet and learning materials to make our project complete.

## CHAPTER 12 – FUTURE SCOPE

The project assists well to record the income and expenses in general. However, this project has some limitations:

1. The application is unable to maintain the backup of data once it is uninstalled.
2. This application does not provide higher decision capability.

To further enhance the capability of this application, we recommend the following features to be incorporated into the system:

1. Multiple language interface.
2. Provide backup and recovery of data.

- 3. Provide better user interface for user.
- 4. Mobile apps advantage.

## **CHAPTER 13 – APPENDIX**

### **GITHUB LINK:**

**<https://github.com/IBM-EPBL/IBM-Project-50145-1660895709>**

### **PROJECT DEMO LINK:**

**<https://drive.google.com/file/d/1qA5lR8r7a2LiN5cSFN2MB5kOHutg5bHQ/view?usp=sharing>**

